CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

Order Picking Problems under Weight, Fragility and Category Constraints

Thomas Chabot
Rahma Lahyani
Leandro C. Coelho
Jacques Renaud

September 2015

CIRRELT-2015-49

Document de travail également publié par la Faculté des sciences de l'administration de l'Université Laval, sous le numéro FSA-2015-012.

Bureaux de Montréal :
Université de Montréal
Pavillon André-Aisenstadt
C.P. 6128, succursale Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

Université Laval · McGill · Concordia · ÉTS · UQÀM · HEC Montréal · Polytechnique Montréal · Université de Montréal

Order Picking Problems under Weight, Fragility and Category Constraints

Thomas Chabot[1,*], Rahma Lahyani[2], Leandro C. Coelho[1], Jacques Renaud[1]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Operations and Decision Systems, 2325 de la Terrasse, Université Laval, Québec, Canada G1V 0A6

[2] College of Business, Al Faisal University, Kingdom of Saudi Arabia

Abstract. Warehouse order picking activities are among the ones that impact the most the bottom lines of warehouses. They are known to account for more than half of the total warehousing costs. New practices and innovations generate new challenges for managers and open new research avenues. Many practical constraints arising in real-life have often been neglected in the scientific literature. In this paper we incorporate many product specific characteristics such as weight, fragility, and category of the items. Motivated by our observation of a real-case, we introduce the order picking problem underweight, fragility and category constraints. We make a full description of the warehouse which enables us to algebraically compute the distances between all pairs of products. We then propose two mathematical models to formulate the problem, and we present four heuristic methods, including extensions of the classical largest gap, midpoint, and S-shape heuristics. The fourth one is an implementation of the powerful adaptive large neighborhood search algorithm specifically designed for the problem at hand. On a set of instances generated from our real-life observations, we assess the performance of all our methods and derive some important insights for managing order picking in this kind of warehouses.

Keywords. Order-picking, warehousing, formulation, exact algorithms, heuristic methods.

Acknowledgements. This research was partly supported by grants 2014-05764 and 0172633 from the Natural Sciences and Engineering Research Council of Canada (NSERC). This support is gratefully acknowledged. We also thank Calcul Québec for providing computing facilities.

Results and views expressed in this publication are the sole responsibility of the authors and do not necessarily reflect those of CIRRELT.

Les résultats et opinions contenus dans cette publication ne reflètent pas nécessairement la position du CIRRELT et n'engagent pas sa responsabilité.

---

* Corresponding author: Thomas.Chabot@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2015

© Chabot, Lahyani, Coelho, Renaud and CIRRELT, 2015

1 Introduction

Order picking represents an important part of warehouse and inventory management activities [10]. It consists of retrieving goods from specific storage locations to fulfill customers orders. It is considered as the most labor-intensive and costly warehousing activity [8, 31], and it may account for up to 55% of all warehouse operating expenses [5]. Therefore, warehouse productivity is highly affected by order picking activities [8]. Research in this area has rapidly expanded over the past decades. These studies can be categorized into books [13, 1, 18], survey papers [33, 8, 12], and theoretical papers providing mathematical formulations and exact or approximate solution methods [2, 16].

Order picking strategies and algorithms have often been studied for classical warehouses [23, 24]. The role of picker personality in predicting picking performance with different picking technologies has been studied by de Vries et al. [9]. Recently, some attention has been devoted to researches oriented towards more realistic picking contexts [4, 20, 3]. These cases are either motivated by the complex characteristics of real-life warehousing activities, legal regulations, or the introduction of on-line shopping requiring faster and more customized services.

This paper is motivated by the situation prevailing in the grocery retail industry. In this industry, each company owns one or several distribution centers (DCs) to store its products. Almost all DCs are designed with reserve storage and fast pick areas. In the fast pick area, each stock keeping unit (SKU) is stored in a specific and dedicated location. However, in the reserve storage area, SKUs are placed at random available locations, but close to the picking and sorting area. Order pickers only have access to the pick area, which is replenished continually by employees from the reserve area. Each order is picked in a sequential way and transported to the sorting area by hand-lift trucks. Each picker deals with a list of products, i.e., a pick-list, to be collected from the storage aisles.

The sequences followed by the pickers to retrieve all the SKUs of a given order are called routes. A fixed routing plan may perform well for some pick-lists, but poorly for others

as decisions are made sequentially. Thus, the primary and most common objective in manual order-picking systems is to find the set of picking routes that minimizes the total traveled distance [6, 7, 25].

In this paper we develop order picking routes in a warehouse under practical restrictions observed in the grocery retail industry. Each product characteristic considered adds an extra layer of difficulty to the problem. These product-specific properties are described next.

We introduce the order picking problem under weight, fragility and category constraints (OPP-WFCC), thus incorporating three new features. Regarding the weight, as soon as the total weight of all the products transported on the hand-lift truck exceeds a threshold value, *heavy items* can no longer be collected, and the picker is only allowed to pick *light* SKUs. Thus, another tour is required to pick (some of) the remaining heavy items. This requirement, referred to as *weight* constraint has two motivations: it helps avoid work accidents and back problems caused by lifting heavy charges to relatively high positions, and it ensures the vertical and horizontal stability of the pallet. Besides the weight, other constraints arise in practice regarding the fragility of the items. A product can support a certain weight without being crushed. Therefore, light or fragile products must not be placed underneath heavy products, i.e., heavy products must be on the bottom of the pallet and light products on the top. This is closely related to the stability of the pallet. We refer to this restriction as *fragility* constraint, and the maximum weight a product can support is referred to hereinafter as *self-capacity*. Finally, we consider two types of commodities: food and non-food products. Non-food products encompass household items. Food products should not be carried under non-food on the pallet in order to avoid contamination. Thus, one has to pick the non-food products separately or before any food products. This constraint is referred to as *category* constraint.

Order picking problems dealing with the physical properties of the products have not been widely studied from a scientific perspective [19], but similar constraints appear in other contexts. Matusiak et al. [20] refer to these constraints as precedence constraints

since they impose that some products must be picked before some others due to weight restrictions, fragility, shape, size, stackability, and preferred unloading sequence. They propose an heuristic method to solve the joint order batching and picker routing problem without any specific assumption regarding the layout and without any pre-determined sequencing constraints. Junqueira et al. [14] introduce the problem of loading rectangular boxes into containers while considering cargo stability, and load bearing constraints. Their mathematical model ensures the vertical and horizontal cargo stability and limits the maximum number of boxes that can be loaded above each other.

The contributions of this paper are threefold. First, we introduce a rich variation of the order picking problem under weight, fragility and category constraints inspired from the grocery retail industry. Part of the complexity of the problem is due to the new practical constraints arising from the separation of the products, their fragility, the stability, and weight limit of the pallet. The second contribution is the development two distinct formulations to model the problem, which include a precise computation of the distance matrix between all pairs of items within the warehouse. Our third contribution is to develop heuristic algorithms and different exact methods to support warehouse picking operations in choosing the most suitable routing sequences to satisfy orders. Specifically, we propose branch-and-cut as well as an adaptive large neighborhood search (ALNS) and an extension of three classical order picking algorithms to solve the OPP-WFCC. We then solve large sets of instances reproducing realistic configurations using our algorithms, which aim at minimizing the total distance traveled for picking all items.

The remainder of the paper is organized as follows. In Section 2 we formally describe the problem and define its particularities. Section 3 presents two mathematical models along with a set of new valid inequalities. The details of the four heuristic algorithms and of the branch-and-cut procedures are provided in Section 4. The results of extensive computational experiments are presented in Section 5, and our conclusions follow in Section 6.

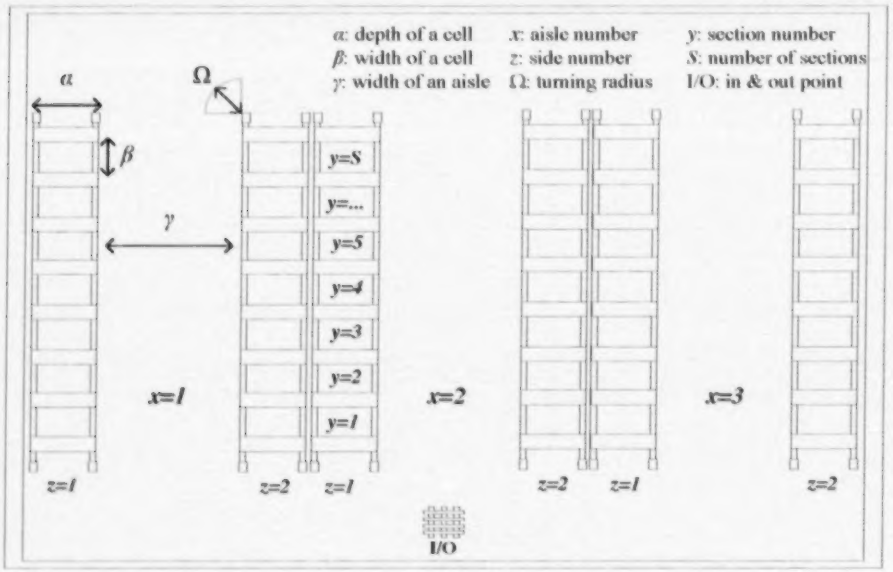
2 Problem description and distance modeling

In order to properly model the problem, which aims to minimize the total distance traveled by the pickers, one needs to know precisely the distances between all pairs of positions within the warehouse under study. Whave modeled a generic warehouse constituted of several parallel aisles. There are complete racks in the middle of the warehouse and two half-racks on either side. These aisles are perpendicular to two corridors, one in the front and one in the rear. This configuration is commonly used and is generic enough for the purpose of this paper. Besides, the location of a product is fixed and each product takes just one location. In other words, each product exists in only one location within the warehouse. Finally, we suppose that the distance between the two sides of an aisle is large, such that the picker cannot pick items from both sides at the same time. The warehouse has one input/output (I/O) point which is considered as the departure and the arrival point for all the pickers. In addition, the warehouse is symmetric with respect to the I/O, with the same number of aisles on either side.

The distance between different locations is computed by solving a shortest path problem. Observe that in order to change aisles, the picker can move through the rear or the front corridor, and these two paths usually yield different distances. Let x_i represent the aisle number of SKU i, and y_i its section number, where $y_i \in \{1, \ldots, S\}$. The total number of sections S represents the number of in-depth locations of the aisle. A warehouse with one aisle and 20 sections per aisle contains 40 locations, by considering both sides. Note that aisles with multiple stages or floors are easily modeled as well, but it is not a common layout in the grocery industry. The cells on the upper level are regularly used for filling the cells at ground level. The side of product i within an aisle is z_i, with $z_i \in \{1, 2\}$. Using this notation, one can fully represent the location of product i as its coordinates (x_i, y_i, z_i) [3]. The width of an aisle is γ units of distance, the depth of a cell is α units of distance, and its width is β. Since there are S sections in an aisle, the total length of the aisle is βS. The minimal distance between two aisles is given by 2α. In order to consider the complex characteristics of the real-world application under study, we consider a turning

radius, denoted Ω. These features can be visualized in Figure 1.

Figure 1: Overview of the generic warehouse in the grocery retail industry



We define the distance matrix $\mathcal{D} = d_{ij}$ where $i < j$ and $i \in \{0, \ldots, m-1\}$, $j \in \{1, \ldots, m\}$, over the I/O point and all m product locations. Several scenarios must be considered when defining $\mathcal{D}$. If the two locations i and j refer to products, two cases arise. The first one appears when both products are in the same aisle. The distance between i and j is then:

$$d_{ij} = |y_i - y_j|\,\beta + |z_i - z_j|\,\gamma \quad \text{if } x_i = x_j. \tag{1}$$

The first term computes the distance between the two sections, and the second term accounts for the aisle width γ if the products are on different sides of the aisle.

The second case arises when the two products are in different aisles. With the aim of easing the notation, we separate the equations into two segments: the vertical distance, denoted v and the horizontal distance described next. The vertical proportion is expressed by $v = \min(\beta(2S - y_i - y_j), \beta(y_i + y_j)) + 2\Omega$. The total distance can then be computed as:

$$d_{\substack{ij \\ i<j}} = \begin{cases} |x_i - x_j|\,(2\alpha + \gamma) + v & \text{if } z_i = z_j \quad (2a) \\ (x_j - x_i)(2\alpha + \gamma) + \gamma + v & \text{if } z_i = 1, z_j = 2 \quad (2b) \\ (x_j - x_i)(2\alpha) + (x_j - (x_i + 1)) + v & \text{if } z_i = 2, z_j = 1. \quad (2c) \end{cases}$$

The first case appears when products are on same side, but in different aisles. In the second case, the items are in two different sides, such that the picker has to cross the aisle width at departure and at the arrival aisles. The third case, product i and j are placed such that the start and arrival aisles are not crossed.

The distances between a product and the I/O point are symmetric and must be computed by taking into account three cases. The first and easiest one is when section x_i of item i coincides with section x_0 of the I/O point. The second appears if $x_i < x_0$, and the third if $x_i > x_j$. These distances are then computed as:

$$d_{i0} = \begin{cases} y_i\beta + \frac{1}{2}\gamma & \text{if } x_i = x_0 \\ (x_0 - x_i)2\alpha + (x_0 - x_i - z_i + 1)\gamma + \frac{1}{2}\gamma + \Omega + y_i\beta & \text{if } x_i < x_0 \\ (x_i - x_0)2\alpha + (x_i - x_0 + z_i - 2)\gamma + \frac{1}{2}\gamma + \Omega + y_i\beta & \text{if } x_i > x_0. \end{cases} \quad (3)$$

In the first case, in which $x_i = x_0$, the distance is the total of the number of sections plus half of the width of an aisle. The second case appears when the product is located on the left of the I/O point, and the third case if the product is on the right. These two cases are symmetrical and are composed of the width of the cells to be traversed, the width of the aisles to be crossed, the turning radius distance, and the length of the sections needed to traverse to reach the product.

The OPP-WFCC is formally defined on a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ where $\mathcal{V}$ is the vertex set and $\mathcal{A}$ is the arc set. The set $\mathcal{V} = \{0, \ldots, m\}$ contains the location of the I/O point and the m products locations constituting the set $\mathcal{V}' = \{1, \ldots, m\}$. The set $\mathcal{A} = \{(i,j) : i \in \mathcal{V}, j \in \mathcal{V}', i \neq j\}$ is the arc set. For each product $i \in \mathcal{V}'$, let q_i be its the weight. In the OPP-WFCC products have three important characteristics. First,

a product is said to be light if its weight is under B units, otherwise it is considered as a heavy item. Second, a product can also be fragile or non-fragile. Fragile item i is associated with a self-capacity w_i, and for non-fragile items w_i is assumed to be Q which corresponds to the total weight that can be transported on a pallet. Finally, products are also categorized as food or non-food items. Non-food items cannot be loaded on top of food items. With this requirement, we observe that arcs (i, j) with i being a food item and j being a non-food item can be removed from $\mathcal{A}$. Finally, when the total weight of all picked items on the pallet reaches a limit L, no more heavy products can be picked in the same tour. The objective of OPP-WFCC is to find the set of tours minimizing the total distance while respecting the weight, fragility and category constraints.

3 Mathematical formulations

We now provide two different formulations for the OPP-WFCC. In Section 3.1 we present a capacity-indexed formulation which makes explicit the remaining capacity of the pallet traversing each each arc. In Section 3.2 we present a two-index vehicle flow formulation.

Let c_{ij} denote the distance between two nodes i and j defining arc (i, j), and $\mathcal{V}'_h$ be the set of nodes associated with heavy products.

3.1 Capacity indexed formulation

In this section, we propose a new formulation to model the OPP-WFCC, referred to as capacity-indexed formulation. This type of formulation has only appeared a few times for basic variants of VRPs [26, 22, 27, 15]. This formulation is compact enough to enumerate all variables and constraints for small and medium instances of the problem. We later incorporate new procedures to further reduce the number of variables. Let binary variables x_{ij}^q indicate that arc (i, j) is used with a remaining capacity of q units. We define the set $\mathcal{C}$ containing the possible values of index q. More information about the generation of

possible values will be described in Section 4.2.1. The formulation is defined by:

$$\text{(F1) minimize} \sum_{(i,j)\in\mathcal{A}} c_{ij} \sum_{q\in\mathcal{C}} x_{ij}^{q} \tag{4}$$

subject to

$$\sum_{j\in\mathcal{V}} \sum_{q\in\mathcal{C}} x_{ij}^{q} = 1 \quad i \in \mathcal{V} \tag{5}$$

$$\sum_{i\in\mathcal{V}} \sum_{q\in\mathcal{C}} x_{ij}^{q} = 1 \quad j \in \mathcal{V} \tag{6}$$

$$\sum_{i\in\mathcal{V}} x_{ij}^{q} - \sum_{k\in\mathcal{V}} x_{jk}^{q-q_j} = 0 \quad j \in \mathcal{V}', q > q_i \in \mathcal{C} \tag{7}$$

$$\sum_{i\in\mathcal{V}'} x_{i0}^{q} = 0 \quad q > 1 \in \mathcal{C} \tag{8}$$

$$\sum_{j\in\mathcal{V}'} x_{0j}^{0} = 0 \tag{9}$$

$$x_{ij}^{q} \in \{0,1\} \quad i,j \in \mathcal{V}, q \in \mathcal{C}. \tag{10}$$

The traveled distance is minimized in (4). Equations (5) and (6) are the in and out degree constraints. They ensure that each node is visited exactly once. Equations (7) guarantee that if the picker visits a node j with a remaining capacity q, then he must leave this node after picking load q_j with a remaining capacity of $q - q_j$. These constraints ensure the connectivity of the solution and the capacity requirements. Infeasible and unnecessary variables are removed with equalities (8) and (9). Equations (8) forbid arcs to return to the I/O point with a remaining capacity different from zero. Similarly, equations (9) state that all arcs visiting a node i must carry a load with available space for item i. Constraints 10 define the domain of the variables.

The main advantage of formulation F1 is that it enables to preprocess the model to decrease its size and to remove infeasible variables by adding appropriate constraints in the preprocessing of the variables. Indeed, because of the physical characteristics of the products, precedence constraints should be considered when defining picking route to avoid

putting non-food products above food products on the pallet. One can identify arcs going from non-food to food products beforehand. Moreover, one can ensure that the safety requirements for fragile products are always respected by removing infeasible variables. Because the index q carries the information about the remaining weight capacity, all arcs with q greater than the self-capacity w_j and the weight q_j of the destination product j, i.e., $q > w_j + q_j$, cannot exist. Thus, all the arcs generated respect the weights that fragile products can support. We can also eliminate variables x_{ij}^q traversing arcs (i, j) with infeasible remaining capacity, i.e., with a capacity $q < Q - q_i$. In what concerns the *weight* constraint, the picker cannot load any heavy product if the total weight of the pallet reaches the threshold L. We model this restriction with inequalities (11):

$$\sum_{i,j \in \mathcal{S}} x_{ij}^q + \sum_{h \in \mathcal{V}_h' \setminus \mathcal{S}} x_{lh}^q \leq |\mathcal{S}| - 1 \quad \mathcal{S} \subseteq \mathcal{V}' : q_s \geq L, l \in \mathcal{S}. \tag{11}$$

Recall that a heavy product is a product whose weight exceeds the limit B. Consider a subset of nodes $\mathcal{S}$ associated with products whose total weight is greater than or equal to L. We check if the potential item to be picked in the same route is a heavy product and we impose that the picker can only pick light items. The arc (l, h) links the last node l of the subset $\mathcal{S}$ and the node corresponding to the heavy product h. Constraints (11) can be improved by considering all the potential heavy products that cannot be picked on the same route. We lift the second term of the left-hand side over the nodes related to heavy products, i.e., the nodes in the subset $\mathcal{V}_h'$. Constraints (11) can thus be replaced by:

$$\sum_{i,j \in \mathcal{S}} x_{ij}^q + \sum_{h \in \mathcal{V}_h' \setminus \mathcal{S}} x_{lh}^q \leq |\mathcal{S}| - 1 \quad \mathcal{S} \subseteq \mathcal{V}' : q_s \geq L, l \in \mathcal{S}. \tag{12}$$

Constraints (12) can be further generalized by removing the restriction to leave the subset $\mathcal{S}$ from the last node visited in this subset. We then replace constraints (12) by:

$$\sum_{i,j \in \mathcal{S}} x_{ij}^q + \sum_{l \in \mathcal{S}} \sum_{h \in \mathcal{V}_h' \setminus \mathcal{S}} x_{lh}^q \leq |\mathcal{S}| - 1 \quad \mathcal{S} \subseteq \mathcal{V}' : q_s \geq L. \tag{13}$$

Since the number of constraints (13) is exponential (in the order of $O(2^m)$), they cannot be generated a priori. We propose a branch-and-cut algorithm to add them dynamically. Details about this algorithm are provided in Section 4.2.

It is possible to strengthen the formulation by adding some valid inequalities. Taking into account the physical characteristics of the products and their structural capacity, we can identify general situations in which a solution is not feasible. We already use several of these characteristics in the variables creation, and we could derive the following new cuts. In constraints (14), we consider an arc between product i and j, and its self-capacity w_i. If a route follows the path from i to j, and from j to k, then one can impose the following constraints:

$$x_{ij}^{q} + \sum_{\substack{k \in \mathcal{V}' \\ w_i < q_j + q_k}} x_{jk}^{q-q_j} \leq 1 \quad i, j \in \mathcal{V}', q \geq q_j \in \mathcal{C}. \tag{14}$$

We can also use the self-capacity to derive a new family of valid inequalities. Let $\mathcal{S}_1$ represent a set of fragile products and its complement denoted $\mathcal{S}_2$, and all arcs going from $i \in \mathcal{S}_1$ to any product j. We know that all arcs going out of j to a product k with the sum of weights $q_k + q_j$ is larger than the maximum self-capacity of the products in $\mathcal{S}_1$ will not be permitted. So all other self-capacity of any products in $\mathcal{S}_1$ are also violated. The following valid inequality thus forbids this situation:

$$\sum_{i \in \mathcal{S}_1} x_{ij}^{q} + \sum_{\substack{k \in \mathcal{S}_2 \\ \max_{i \in \mathcal{S}_1}\{w_i\} < q_k + q_j}} x_{jk}^{q-q(j)} \leq 1 \quad j \in \mathcal{V}', \mathcal{S}_1, \mathcal{S}_2 \subseteq \mathcal{V}', q \geq q_j \in \mathcal{C}. \tag{15}$$

3.2 Two-index flow formulation

Formulation F1 has the drawback that the number of variables is dependent on the number of q values that can be obtained by different picking routes. We now provide a two-index flow formulation to determine the best picker routes, based on the model described in

Toth and Vigo [32] for the asymmetric VRP. We define new binary variables x_{ij} equal to one if arc (i, j) is used, and an integer variable k indicating the number of picking tours required to satisfy all the orders. This model can be stated as follows:

$$\text{(F2) minimize} \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} \tag{16}$$

subject to

$$\sum_{j\in\mathcal{V}'} x_{0j} = k \tag{17}$$

$$\sum_{i\in\mathcal{V}'} x_{i0} = k \tag{18}$$

$$\sum_{i\in\mathcal{V}} x_{ij} = 1 \quad j \in \mathcal{V}' \tag{19}$$

$$\sum_{j\in\mathcal{V}} x_{ij} = 1 \quad i \in \mathcal{V}' \tag{20}$$

$$\sum_{i,j\in\mathcal{S}} x_{ij} \leq |\mathcal{S}| - r(s) \quad \mathcal{S} \subseteq \mathcal{V}', |\mathcal{S}| \geq 2 \tag{21}$$

$$k \in \mathbb{N}_+ \tag{22}$$

$$x_{ij} \in \{0, 1\} \quad i, j \in \mathcal{V} \tag{23}$$

The objective function (16) minimizes the total traveled distance. Constraints (17) and (18) impose the degree requirements for the I/O point. Equations (19) and (20) are in-degree and out-degree constraints. They state that each product i must be picked exactly once. Constraints (21) correspond to generalized subtour elimination constraints. They simultaneously forbid subtours and ensure the capacity requirements with $r(s) = \left\lceil \frac{\sum_{j\in S} q_j}{Q} \right\rceil$. Constraints (22) and (23) impose integrality and binary conditions on the variables.

Similarly to the capacity-indexed formulation, one can add beforehand the *category* constraints when creating the variables. We also remove irrelevant arcs going from non-food to food products. However, unlike formulation F1, formulation F2 cannot handle the *fragility* constraints in a preprocessing phase by eliminating infeasible variables. Thus, we

propose a branch-and-cut routine that dynamically adds the fragility constraints defined by (24). For a subset of nodes $\mathcal{S}$, we check whether the products related to the potential nodes that may follow node i respect the maximum weight supported by i, w_i. If this condition is not met, we then impose the fragility constraints:

$$\sum_{j \in \mathcal{S}} x_{ij} + \sum_{j,k \in \mathcal{S}} x_{jk} \leq |\mathcal{S}| - 1 \quad \mathcal{S} \subseteq \mathcal{V}^{'} : q_s > w_i, i \in \mathcal{V}^{'} \backslash \mathcal{S}. \tag{24}$$

The *weight* constraints (13) proposed for formulation F1 can be adapted for F2 by removing the capacity index, which results in:

$$\sum_{i,j \in \mathcal{S}} x_{ij} + \sum_{l \in \mathcal{S}} \sum_{h \in \mathcal{V}_h^{'} \backslash \mathcal{S}} x_{lh} \leq |\mathcal{S}| - 1 \quad \mathcal{S} \subseteq \mathcal{V}^{'} : q_s \geq L. \tag{25}$$

We have also adapted the two sets of valid inequalities from model F1 to model F2. They are written as follows:

$$x_{ij} + \sum_{\substack{k \in \mathcal{V}^{'} \\ w_i < q_j + q_k}} x_{jk} \leq 1 \quad i, j \in \mathcal{V}^{'} \tag{26}$$

$$\sum_{i \in \mathcal{S}_1} x_{ij} + \sum_{\substack{k \in \mathcal{S}_2 \\ \max_{i \in \mathcal{S}_1}\{w_i\} < q_k + q_j}} x_{jk} \leq 1 \quad j \in \mathcal{V}^{'} \mathcal{S}_1, \mathcal{S}_2 \subseteq \mathcal{V}^{'}. \tag{27}$$

4 Solution algorithms

In this section we describe several algorithms we have developed and used to solve the OPP-WFCC. We describe four heuristic algorithms in Section 4.1, and the framework of the exact algorithms we have used in Section 4.2.

4.1 Heuristic algorithms

In this section we describe four heuristic algorithms that we propose and use to solve the problem at hand. The first three are extensions of classical and well-known heuristics that we have adapted for the OPP-WFCC, and the last one is a powerful metaheuristic designed specifically for our problem. In Section 4.1.1 we describe how we have adapted the S-shape, mid point, and largest gap heuristics of Hall [11] for our problem. They should be modified to handle the new constraints of our problem. Finally, in Section 4.1.2 we present the ALNS metaheuristic we develop for our problem.

4.1.1 Classical heuristics

Under the S-shape heuristic, the picker will completely traverse any aisle containing at least one item to be picked. Thus, the warehouse is completely traversed, by leaving an aisle and entering the adjacent one, picking products as the picker advances.

In the mid point heuristic we divide the warehouse into two halves. There are picks on the front side, reached by the front aisle, and picks on the back side reached from the back side. The picker only crosses through the first and last aisles of the picking tour. Intermediate aisles are never crossed completely. According to Hall [11], this method performs better than the S-shape when the number of picks per aisle is small.

In the largest gap heuristic, the picker follows the same idea of the S-shape by visiting all adjacent aisles, but instead of traversing the aisle completely, he enters the aisle up to the item that will leave the largest gap for the next item in that aisle. This heuristic tries to maximize the parts of the aisles that are *not* traversed. When the largest gap is identified, the picker returns and leaves the aisle from the same side (back or front) used to enter it. We proceed by making the first pick as in the original heuristic and by traversing the warehouse following the original rules.

The following modifications are required to these three heuristics in order to handle weight, fragility and category constraints. We start by picking the first available product in the

first aisle as in the original heuristics. For any subsequent products, it is picked if it respects all the constraints of the problem, and skipped otherwise. We continue following the heuristic rules traversing the warehousing, and skipping any infeasible pick until the pallet is full or the last item is reached. At this point, the picker returns to the I/O point and starts a new route going back to the first skipped item, which will be picked and the same procedure is repeated as long as there are products to be picked.

4.1.2 Adaptive large neighborhood search heuristic

We present an implementation of an ALNS heuristic, widely based on Ropke and Pisinger [29]. The ALNS is composed of a set of simple destruction and reconstruction heuristics in order to find better solutions at each iteration. An initial solution can be considered to speed up the search and the convergence of the algorithm. We have implemented a fast sequential insertion heuristic, which performs a greedy search for the best insertion for one product at a time.

The ALNS selects one of many destroy and repair operators at each iteration. We have implemented three destroy and two repair operators. Our destroy operators include the Shaw removal [30], the worst removal, and a random removal. Our repair operators include a greedy parallel insertion and a k-regret heuristic [28]. Each operator is selected with a probability that depends on its past performance, and a simulated annealing acceptance criterion is used. Each procedure of the ALNS is adapted to deal with the weight, fragility and category constraints.

The ALNS is ran for 50 000 iterations of destroy-repair operations. After every 100 iterations the weight of each operator is updated according to its past performance. Initially, all the operators have the same weight. A sketch of our ALNS implementation is provided in Algorithm 1 and for further details see Ropke and Pisinger [29].

Algorithm 1 Adaptive large neighborhood search

1: Create an initial solution s using sequential insertion operators : $s' = s$
2: D : set of removal operators, I : set of repair operators
3: Initiate probability ρ^d for each destroy operator $d \in D$ and probability ρ^i for each repair operator $i \in I$
4: **while** stop criterion is not met **do**
5: Select remove ($d \in D$) and insert ($i \in I$) operators using ρ^d and ρ^i
6: Apply operators and create s^t
7: **if** s^t, is accepted by the simulated annealing criterion **then**
8: $s = s^t$
9: **end if**
10: **if** $cost(s) < cost(s')$ **then**
11: $s' = s$
12: **end if**
13: Update ρ^d and ρ^i
14: **end while**
15: return s'

4.2 Exact algorithms

In this section we present the various algorithms used to solve exactly the mathematical models from Section 3. We present in Section 4.2.1 a subset sum algorithm that enables us to drastically reduce the number of required variables for model F1. This is followed in Section 4.2.2 by an overview of the branch-and-cut algorithm we have used as well as the detailed description of the procedures used to dynamically identify and generate violated valid inequalities.

4.2.1 Subset sum to reduce the number of variables of model F1

The variables of formulation F1 presented in Section 3.1 can only be fully enumerated for small and medium size instances. However, it is easy to observe that some variables are never used in the model, e.g., the ones for which some values of q cannot be obtained by any combination of the weights of the products. These variables can be generated and fed into the solver, which will set them to zero in any feasible solution. If one can identify these variables beforehand, it is possible to set them to zero and remove them from the model at a preprocessing phase. Thus, one can (substantially) decrease the size of the model and the memory footprint by preprocessing the model and the instance a priori, identifying the subset of variables that should not be generated.

We use a subset sum algorithm to identify all possible values of q from 1 to Q that can be achieved by any combination of weights q_i. The ones that are found not to be feasible are not generated.

From a theoretical point of view, its performance is directly related to the distribution of the weights of the items in the instance. For example, an instance for which all products have a weight of five units will have five time less variables than the model with all the values of q.

4.2.2 Branch-and-cut algorithms

Models F1 and F2 can be fed straightforwardly into a general purpose solver and solutions are obtained by branch-and-bound if the number of constraints (13), (21), (24) and (25) is not excessive. However, for instances of realistic size, the number of these constraints is too large to allow a full enumeration and they must be dynamically generated throughout the search process. Indeed, polynomial constraints may be added a priori to the model while other constraints cannot be generated a priori since their number is exponential.

The exact algorithm we present is a branch-and-cut scheme in which inequalities (13), (21), (24) and (25) are generated and added into the program whenever they are found to be violated. It works as follows. At a generic node of the search tree, a linear program containing the model with a subset of the subtour elimination constraints and relaxed integrality constraints is solved, a search for violated inequalities is performed, and some of these are added to the current program which is then reoptimized. This process is reiterated until a feasible or dominated solution is reached, or until there are no more cuts to be added. At this point, branching on a fractional variable occurs. We provide a sketch of the branch-and-cut scheme in Algorithm 2. Note that for formulation F1, we apply Algorithm 2 but we skip the process to identify violated subtour elimination constraints (lines $8-12$) as connectivity requirements are ensured by constraints (7).

In this branch-and-cut algorithm, weight and fragility inequalities are used as cutting planes to strengthen the linear programming relaxation at each node of the branch-and-bound tree. Constraints (13), (24) and (25) cannot be generated a priori since their number is exponential. These are initially relaxed and dynamically generated as cuts as they are found to be violated.

When model F1 without *weight* constraints (13) (similarly with (25) for F2) is solved, two situations can occur. The first one consists of finding an integer solution. Then one can easily verify if the picking tour exceeds the weight limit L or the self-capacity w_i by calculating the cumulative weight of a set $\mathcal{S}$. In the second case, when the solution

Algorithm 2 Branch-and-cut algorithm

Subproblem solution: Solve the LP relaxation of the current node
Termination check:
if there are no more nodes to evaluate **then**
    Stop
else
    Select one node from the branch-and-cut tree
end if
while the solution of the current LP relaxation contains subtours **do**
    Identify connected components as in Padberg and Rinaldi [21]
    Add violated subtour elimination constraints
    Subproblem solution. Solve the LP relaxation of the current node
end while
if the solution contains no disconnected components **then**
    Apply Algorithms 3 and 4 and add violated cuts
end if
if the solution of the current LP relaxation is integer **then**
    Go to the termination check
else
    Branching: branch on one of the fractional variables
    Go to the termination check
end if

is fractional, one can identify connected components by means of the maximum flow algorithm as in Padberg and Rinaldi [21]. This procedure, sketched in Algorithm 3, consists of constructing an auxiliary graph as follows. First, a node i is selected. Then the node j associated with the maximum flow value leaving i is identified and added to the set $\mathcal{S}$. We check whether the sum of weights of the products included in $\mathcal{S}$, referred to by $q_{\mathcal{S}}$, respects the threshold L. If it exceeds this limit, we add cuts to forbid such solution. This is achieved by adding the appropriate constraints (13) for the capacity indexed formulation and (25) for the two-index flow formulation associated with the nodes in $\mathcal{S}$.

Algorithm 3 *Weight* constraints algorithm

1: **for** $i = 1$ to m **do**
2: $\mathcal{S} = \{i\}$
3: $j^* = argmax_{k \in \mathcal{V}' \setminus \mathcal{S}}\{x_{ik}\}$
4: $\mathcal{S} = \mathcal{S} \cup \{j^*\}$
5: **if** $q_{\mathcal{S}} > L$ and $q_{\mathcal{S}} \leq Q$ **then**
6: **for** $l \in \mathcal{S}$ **do**
7: **if** the solution violates (13) (or (25)) **then**
8: Add *weight* constraints (13) (or (25))
9: **end if**
10: **end for**
11: **end if**
12: **if** $q_{\mathcal{S}} > Q$ **then**
13: Continue to next i from step 1
14: **else**
15: Go to Step 3
16: **end if**
17: **end for**

Similarly, in Algorithm 4, we describe the procedure used to dynamically generate constraints (24). We identify a subset $\mathcal{S}$ such that it respects the *fragility* constraints. Oth-

erwise, we add the violated constraints associated with the nodes in $\mathcal{S}$.

Algorithm 4 *Fragility* constraints algorithm

for $i = 1$ to m **do**
  if i is a fragile product **then**
    $\mathcal{S} = \{i\}$, best $= i$
    while stop criterion is not met **do**
      $j^* = argmax_{k \in \mathcal{V}' \setminus \mathcal{S}}\{x_{best,k})\}$
      if $x_{best,k} < 0.5$ **then**
        Stop
      end if
      $\mathcal{S} = \mathcal{S} \cup \{j^*\}$, Best$=j$
      if $q_{\mathcal{S}} \leq Q$ and $q_{\mathcal{S}} > w_i$ **then**
        if the solution violates (24) **then**
          Add *fragility* constraints (24)
        end if
      end if
      if $q_{\mathcal{S}} > Q$ **then**
        Stop
      end if
    end while
  end if
end for

5 Computational experiments

In this section we describe the results of extensive computational experiments performed on all algorithms presented in Section 4, which were all implemented in C++. The branch-and-cut procedure uses the CVRPSEP library [17], our own cutting methods and IBM

CPLEX Concert Technology 12.6 as the branch-and-bound solver. All computations were executed on machines equipped with two Intel Westmere EP X5650 six-core processors running at 2.667 GHz, and with up to 8 GB of RAM installed per node running the Scientific Linux 6.3. All algorithms were provided a time limit of 7200 seconds.

5.1 Instances generation

We have created three groups of instances to represent different configurations and combinations of the new features introduced in this paper. To reflect common practice, non-food items are grouped in the same aisle(s). In the first group of instances, called G1, only one side of the first aisle is dedicated to non-food items, whereas all fragile and non-fragile products are randomly placed throughout the warehouse. In the second group of instances (G2), we split the picking area in two symmetric zones. This allows non-food items to be placed in the lateral extremities of the warehouse. All remaining items are placed randomly elsewhere. In the third group (G3), non-food items are placed in the extremities like in G2, but solid products (SP), i.e. non-fragile, with large self-capacity are grouped together in the sections close to the I/O point. These three types of layouts are illustrated in Figure 2.

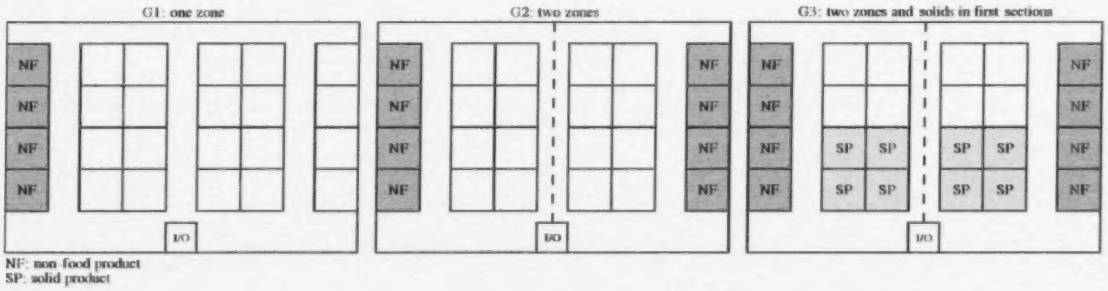


Figure 2: Schematic representation of the three groups of instances

In each group of instances, we have kept the same proportion of food and non-food products, and ranges of weight and self-capacity. The random generator has a 50% probability to create a fragile product. Each fragile product has a 2/3 probability to be a food product

and 1/3 to be non-food. A non-fragile product, food or otherwise, has a weight between five and 50 kg, and infinite self-capacity. A fragile product has a weight between one and 10 kg, and a self-capacity between five and 50 kg.

An instance is characterized by the number of picks, the number of aisles and the capacity of the picking vehicles. The number of SKU to be picked is between 20 and 100, by steps of 10. The number of aisles is three, five, or seven, with the I/O point in the middle of the warehouse. Finally, the capacity of the pallets is 150 or 250 kg. In different groups, only the location of an item changes. For each combination of SKU, number of aisles and capacity, we create three instances, with 162 instances per group, and 486 in total.

5.2 Detailed results

This section presents the results of extensive computational experiments of all our algorithms for all the 486 instances. We start our analysis by comparing the results of the four heuristic algorithms presented in Section 4.1. In Table 1 we present the solutions of the heuristic algorithms for all groups of instances using pallets with $Q = 150$, and in Table 2 for the larger pallets with $Q = 250$. In these tables we show the total distance yielded by the heuristic as well as the number of required routes. We note that among the three classical heuristics, the mid point procedure outperformed by a small margin the largest gap and the S-shape methods. A richer and more intricate heuristics such as the ALNS can significantly improve the solution, requiring less picking tours and much shorter traveled distance. The first three heuristics have a negligible running time, whereas the ALNS can output a very good solution within a few seconds of computing time.

As can be observed from Tables 1 and 2, the three classical heuristics yield very similar solutions, whereas the ALNS could significantly improve it. However, no lower bounds are known to allow us to assess the quality of these solutions. For this reason, we now evaluate the quality of the solutions and lower bounds obtained using exact methods.

In Table 3 we present the results with the capacity $Q = 150$, and for $Q = 250$ in Table 4.

Table 1: Average distance and number of routes for the four heuristic algorithms on instances with $Q = 150$

Groups	# pickups	Classical heuristics						ALNS		
		S-shape		Mid point		Largest gap				
		Distance	Routes	Distance	Routes	Distance	Routes	Distance	Routes	Time (s)
G1	20	1951.7	5.9	1883.9	7.8	2024.4	5.1	1215.6	3.9	3.6
	30	2462.8	8.0	2493.3	9.1	2515.0	6.4	1561.7	5.0	8.3
	40	3058.9	9.8	2977.2	11.4	3081.1	8.7	1850.6	6.8	18.4
	50	3716.7	12.6	3700.6	14.0	3782.2	10.0	2141.1	7.4	33.1
	60	4240.6	13.9	4044.4	14.9	4135.6	11.3	2496.1	8.9	56.3
	70	4597.8	15.8	4475.6	16.4	4710.6	13.0	2728.9	10.2	78.6
	80	5062.2	17.9	4950.6	18.4	5166.1	15.2	3046.7	11.7	118.3
	90	5716.7	20.3	5876.1	22.0	5934.4	17.8	3572.8	14.1	170.0
	100	5896.7	22.1	5594.4	21.3	5798.3	18.6	3607.2	14.9	232.4
	Average	**4078.2**	**14.0**	**3999.6**	**15.1**	**4127.5**	**11.8**	**2469.0**	**9.2**	**79.9**
G2	20	1948.3	6.4	1950.0	7.4	2021.7	5.6	1252.8	4.0	3.7
	30	2708.9	8.9	2540.0	9.6	2720.0	7.2	1666.7	5.4	8.2
	40	2979.4	10.3	2908.9	11.1	3043.9	9.0	1926.1	6.9	17.1
	50	3463.3	12.3	3525.6	13.3	3603.3	10.9	2180.0	8.3	30.9
	60	4047.2	13.7	3948.3	14.6	4085.6	12.0	2468.3	9.3	54.0
	70	4601.1	16.9	4518.9	16.9	4732.8	14.3	2842.8	10.4	79.0
	80	4998.3	17.7	4888.3	18.2	5066.7	15.8	3050.0	11.7	121.1
	90	5877.8	20.8	5640.6	21.2	5834.4	17.8	3666.1	14.4	180.7
	100	6028.9	22.2	5841.1	22.3	6117.8	19.0	3678.9	14.9	221.6
	Average	**4072.6**	**14.4**	**3973.5**	**15.0**	**4136.2**	**12.4**	**2525.7**	**9.5**	**79.6**
G3	20	1855.0	6.2	1858.9	7.1	1889.4	5.6	1263.9	4.1	3.2
	30	2536.7	8.7	2481.7	9.2	2623.9	7.1	1612.2	5.4	8.4
	40	2984.4	10.2	2924.4	10.7	3061.1	8.9	1848.3	7.0	17.8
	50	3458.3	12.3	3389.4	12.6	3606.7	10.6	2155.0	8.2	33.7
	60	3998.9	14.1	3766.1	14.2	4060.0	12.1	2368.9	9.3	55.4
	70	4383.9	15.8	4402.8	16.1	4518.3	13.8	2698.3	10.9	77.6
	80	4781.7	16.9	4683.9	17.6	4872.8	15.0	2916.1	12.0	110.3
	90	5415.0	19.2	5432.8	20.0	5591.1	17.2	3372.8	14.7	148.2
	100	5612.8	20.7	5435.0	20.7	5707.2	17.9	3442.8	15.4	200.8
	Average	**3891.9**	**13.8**	**3819.4**	**14.2**	**3992.3**	**12.0**	**2408.7**	**9.7**	**72.8**
Global average		**4014.2**	**14.1**	**3930.8**	**14.8**	**4085.4**	**12.1**	**2467.8**	**9.5**	**77.4**

Table 2: Average distance and number of routes for the four heuristic algorithms on instances with $Q = 250$

Groups	# pickups	Classical heuristics						ALNS		
		S-shape		Mid point		Largest gap				
		Distance	Routes	Distance	Routes	Distance	Routes	Distance	Routes	Time (s)
G1	20	1840.0	5.2	1748.3	6.9	1708.9	4.1	1114.4	3.1	4.3
	30	2453.9	7.6	2523.9	9.3	2311.7	5.1	1365.0	3.8	10.0
	40	2996.7	10.3	2859.4	10.9	2778.3	7.7	1656.7	4.8	20.2
	50	3210.6	10.9	3107.8	11.9	3166.1	8.1	1735.6	5.3	39.7
	60	3648.3	12.7	3538.9	12.7	3443.3	9.2	2016.7	6.6	63.9
	70	3988.9	13.3	3760.0	13.7	4013.9	10.6	2146.1	6.9	83.1
	80	4385.6	14.9	4385.6	16.3	4415.6	13.0	2415.6	8.3	129.4
	90	4870.6	17.1	4602.8	16.9	4772.8	13.2	2569.4	8.7	194.0
	100	5556.1	19.1	5134.4	18.7	5174.4	14.2	2835.6	9.9	253.6
	Average	**3661.2**	**12.4**	**3517.9**	**13.0**	**3531.7**	**9.5**	**1983.9**	**6.4**	**88.7**
G2	20	1949.4	5.9	1770.6	6.9	1801.7	4.7	1191.7	3.7	4.1
	30	2486.7	7.8	2423.9	9.2	2325.6	6.3	1413.3	4.3	9.9
	40	2838.3	9.7	2872.2	10.6	2929.4	8.4	1683.9	5.1	19.7
	50	3390.6	11.6	3146.1	11.7	3216.7	8.8	1892.2	5.7	36.8
	60	3778.3	13.7	3389.4	12.7	3579.4	10.3	2075.0	6.9	57.9
	70	4174.4	14.2	3865.6	14.2	4111.7	11.7	2222.8	7.3	84.6
	80	4442.2	15.9	4387.8	16.2	4554.4	13.0	2491.1	8.6	129.8
	90	4818.9	17.3	4430.6	16.2	4711.7	13.7	2614.4	9.2	194.1
	100	5039.4	17.8	5066.1	19.1	5337.2	15.9	2881.7	10.2	242.0
	Average	**3657.6**	**12.6**	**3483.6**	**13.0**	**3618.6**	**10.3**	**2051.8**	**6.8**	**86.5**
G3	20	1853.9	5.7	1711.1	6.6	1827.2	5.1	1181.7	3.6	3.8
	30	2284.4	7.0	2183.3	8.0	2262.2	6.2	1401.1	4.3	9.6
	40	2809.4	9.7	2691.7	9.7	2710.6	7.7	1669.4	5.0	20.3
	50	3209.4	11.2	3023.3	11.0	3019.4	8.6	1822.2	5.8	39.7
	60	3687.2	13.0	3392.8	12.3	3510.6	9.9	2024.4	6.9	63.1
	70	4027.8	13.8	4000.6	14.3	4107.8	11.4	2236.7	7.3	87.1
	80	4379.4	15.4	4278.3	15.6	4499.4	12.9	2421.7	8.2	119.0
	90	4607.2	16.0	4495.6	16.1	4671.1	13.6	2535.6	9.0	179.1
	100	5372.8	18.4	4880.6	17.8	5010.0	14.6	2797.8	10.0	239.2
	Average	**3581.3**	**12.3**	**3406.4**	**12.4**	**3513.2**	**10.0**	**2010.1**	**6.7**	**84.5**
Global average		**3633.4**	**12.4**	**3469.3**	**12.8**	**3554.5**	**9.9**	**2015.3**	**6.6**	**86.6**

We add to these results those of the ALNS heuristic, i.e., the best heuristic solution we have obtained. For each of the two formulations, we present the upper bound (UB), the lower bound (LB), the optimality gap in percentage, and the running time.

From Tables 3 and 4, we observe that the results obtained with the ALNS metaheuristic are very close to the best upper bounds yielded by both models. For $Q = 150$, ALNS gives an overall distance of 2467.8 compared to 2466.6 and 2466.5 for F1 and F2. Moreover, the lower bounds obtained demonstrate that these solutions are very tight, and the optimality gap is always less than 10%, and often less than 5%. Thus, the heuristic solutions obtained with the ALNS heuristic in a few seconds are of very good quality. We also note that all solutions of both models use the same number of routes as those of the ALNS.

Both models had problems closing the gap and proving optimality for instances with more than 30 pickups. With respect to the running times over all instances, F2 is faster than F1, with an average of 5009.4 and 6080.0 seconds, respectively.. This is due to the fact that F2 was able to prove optimality for more instances than F1 within the allotted two hours of running time.

Finally, it is important to notice that the subset sum algorithm presented in Section 4.2.1 was able to reduce an average of 12.55% variables compared to a complete enumeration of all possible variables for model F1. In our test bed, we have observed a reduction of up to 23.56% in the number of variables, and a minimum reduction of 5.87%.

Regarding the three groups of instances, we observe that the gap between the upper and lower bounds for F1 is nearly the same for all groups, but decreases slightly from group G1 to G3, a behavior that is opposed to that of F2. It seems more difficult for F2 to solve instances where the products are placed in groups, in such a way as to facilitate the picking work. There are no major differences among groups in terms of the traveled distance. For G2, where non-food products are separated at each side of the warehouse, it seems to yield solutions with larger total distances.

Table 5 presents the number of optimal solutions obtained by both models over the 486

Table 3: Average results of the exact algorithms on instances with $Q = 150$

Group	# pickups	ALNS	Capacity indexed formulation (F1)				Two-index formulation (F2)			
			UB	LB	Gap (%)	Time (s)	UB	LB	Gap (%)	Time (s)
G1	20	1215.6	1213.3	1213.3	0.0	104.0	1213.3	1213.3	0.0	6.4
	30	1561.7	1557.8	1528.7	2.1	3221.1	1556.1	1556.1	0.0	220.8
	40	1850.6	1848.9	1729.1	6.6	6575.7	1840.6	1829.1	0.6	2447.9
	50	2141.1	2141.1	1975.5	7.7	7200.0	2135.6	2116.2	0.9	3066.6
	60	2496.1	2496.1	2292.0	8.4	7200.0	2496.1	2372.6	4.9	6847.2
	70	2728.9	2728.3	2519.1	7.8	7200.0	2728.9	2563.8	6.5	7200.0
	80	3046.7	3046.7	2751.6	9.8	7200.0	3046.7	2835.8	7.1	7200.0
	90	3572.8	3572.8	3306.4	7.6	7200.0	3572.8	3357.7	6.0	7200.0
	100	3607.2	3607.2	3287.8	9.0	7200.0	3607.2	3354.5	7.1	7200.0
	Average	**2469.0**	**2468.0**	**2289.3**	**6.5**	**5900.1**	**2466.4**	**2355.7**	**3.7**	**4598.8**
G2	20	1252.8	1252.8	1194.2	6.1	39.4	1252.8	1252.8	0.0	50.8
	30	1666.7	1665.6	1639.0	1.9	3477.0	1665.6	1665.6	0.0	36.8
	40	1926.1	1924.4	1826.2	5.6	6239.6	1922.2	1905.7	0.9	2671.4
	50	2180.0	2180.0	2045.1	6.3	6553.2	2180.0	2096.6	4.1	5420.2
	60	2468.3	2459.4	2279.3	7.3	7200.0	2468.3	2347.6	5.0	7200.0
	70	2842.8	2842.8	2590.0	9.0	7200.0	2842.8	2626.1	7.6	7200.0
	80	3050.0	3050.0	2787.6	8.6	7200.0	3050.0	2821.0	7.6	7200.0
	90	3666.1	3666.1	3395.8	7.3	7200.0	3666.1	3428.5	6.5	7200.0
	100	3678.9	3678.9	3351.8	8.7	7200.0	3678.9	3426.2	6.8	7200.0
	Average	**2525.7**	**2524.4**	**2345.4**	**6.8**	**5812.1**	**2525.2**	**2396.7**	**4.3**	**4908.8**
G3	20	1263.9	1263.9	1263.9	0.0	74.0	1263.9	1263.9	0.0	3.9
	30	1612.2	1607.8	1575.4	2.0	2975.0	1611.7	1603.5	0.6	1086.0
	40	1848.3	1844.4	1747.4	5.8	5729.1	1847.2	1811.1	2.1	5788.0
	50	2155.0	2150.6	1984.7	7.8	7200.0	2151.1	2001.0	7.3	7200.0
	60	2368.9	2368.9	2189.1	7.5	6675.1	2366.7	2255.8	4.8	7200.0
	70	2698.3	2698.3	2492.7	7.9	7200.0	2698.3	2458.4	9.2	7200.0
	80	2916.1	2916.1	2678.0	8.2	7200.0	2916.1	2662.2	8.8	7200.0
	90	3372.8	3372.8	3129.4	7.3	7200.0	3372.8	3136.3	7.2	7200.0
	100	3442.8	3442.8	3142.8	8.8	7200.0	3442.8	3133.0	9.2	7200.0
	Average	**2408.7**	**2407.3**	**2244.8**	**6.2**	**5717.0**	**2407.8**	**2258.4**	**5.5**	**5564.2**
Global average		**2467.8**	**2466.6**	**2293.2**	**6.5**	**5809.7**	**2466.5**	**2336.8**	**4.5**	**5023.9**

Table 4: Average results of the exact algorithms on instances with $Q = 250$

Group	# pickups	ALNS	Capacity indexed formulation (F1)				Two-index formulation (F2)			
			UB	LB	Gap (%)	Time (s)	UB	LB	Gap (%)	Time (s)
G1	20	1114.4	1111.7	1092.2	1.3	1114.2	1111.7	1111.7	0.0	0.2
	30	1365.0	1351.7	1276.7	5.5	4442.8	1351.7	1351.7	0.0	25.0
	40	1656.7	1656.7	1509.5	8.9	7200.0	1648.9	1637.2	0.9	2606.3
	50	1735.6	1735.6	1511.5	12.7	7200.0	1735.6	1651.3	5.0	5163.6
	60	2016.7	2016.7	1701.2	15.7	7200.0	2016.7	1802.9	11.2	7200.0
	70	2146.1	2146.1	1846.8	13.9	7200.0	2146.1	1944.4	9.9	7200.0
	80	2415.6	2415.6	2086.4	13.8	7200.0	2415.6	2128.1	12.0	7200.0
	90	2569.4	2569.4	2148.2	16.5	7200.0	2569.4	2212.0	13.7	7200.0
	100	2835.6	2835.6	2479.1	12.8	7200.0	2835.6	2462.9	13.3	7200.0
	Average	**1983.9**	**1982.1**	**1739.1**	**11.3**	**6217.4**	**1981.2**	**1812.2**	**7.3**	**4866.1**
G2	20	1191.7	1188.9	1171.3	1.1	1543.9	1188.9	1188.9	0.0	0.4
	30	1413.3	1411.1	1332.0	5.7	5902.1	1410.0	1410.0	0.0	855.1
	40	1683.9	1673.9	1543.1	7.7	7200.0	1682.2	1673.3	0.7	1261.9
	50	1892.2	1892.2	1596.8	15.2	7200.0	1889.4	1691.5	10.5	7200.0
	60	2075.0	2075.0	1730.3	16.3	7200.0	2075.0	1837.0	11.6	7200.0
	70	2222.8	2222.8	1912.7	13.4	7200.0	2222.8	1994.4	10.0	7200.0
	80	2491.1	2491.1	2143.2	14.1	7200.0	2491.1	2181.8	12.9	7200.0
	90	2614.4	2614.4	2216.0	15.3	7200.0	2614.4	2233.4	14.3	7200.0
	100	2881.7	2881.7	2511.9	13.2	7200.0	2879.4	2493.2	13.8	7200.0
	Average	**2051.8**	**2050.1**	**1795.2**	**11.3**	**6427.3**	**2050.4**	**1855.9**	**8.2**	**5035.3**
G3	20	1181.7	1169.4	1169.4	0.0	1689.7	1169.4	1169.4	0.0	0.3
	30	1401.1	1393.3	1306.6	6.3	6164.2	1397.2	1390.8	0.4	885.1
	40	1669.4	1657.2	1551.0	7.0	6602.8	1668.9	1636.8	2.2	2477.0
	50	1822.2	1822.2	1591.0	12.5	7200.0	1810.6	1650.8	9.0	6419.9
	60	2024.4	2024.4	1729.4	14.5	7200.0	2013.9	1801.8	10.8	7166.3
	70	2236.7	2236.7	1907.0	14.6	7200.0	2236.7	1963.2	12.4	7200.0
	80	2421.7	2421.7	2117.5	12.7	7200.0	2421.7	2125.2	12.7	7200.0
	90	2535.6	2535.6	2159.7	14.9	7200.0	2535.6	2154.0	14.9	7200.0
	100	2797.8	2797.8	2464.9	12.0	7200.0	2797.8	2424.9	13.5	7200.0
	Average	**2010.1**	**2006.5**	**1777.4**	**10.5**	**6406.3**	**2005.7**	**1814.1**	**8.4**	**5083.2**
Global average		**2015.2**	**2012.9**	**1770.6**	**11.0**	**6350.4**	**2012.4**	**1826.7**	**8.0**	**4994.9**

instances. These results are separated in groups of instances, number of aisles, and by the value of the capacity Q. This helps highlight the effect of each of these characteristics. In the first and second columns, we present the groups of instances and the number of aisles. We then present the results for the capacity indexed formulation F1 and for the two-index formulation F2. It is clear from this table that for all cases, the two-index formulation outperformed the capacity-indexed formulation by a large margin. This corroborates the results already observed from the gaps and running time of these models.

Table 5: Number of optimal solutions per group of instances and capacity of the truck

	# aisles	# optimal F1 $Q = 150$	# optimal F1 $Q = 250$	# optimal F2 $Q = 150$	# optimal F2 $Q = 250$
G1	3	4	4	11	8
	5	6	4	10	12
	7	6	4	10	9
	Total	**16**	**12**	**31**	**29**
G2	3	4	4	8	8
	5	5	3	10	9
	7	7	3	9	9
	Total	**16**	**10**	**27**	**26**
G3	3	5	4	5	7
	5	7	3	7	9
	7	6	6	8	9
	Total	**18**	**13**	**20**	**25**

6 Conclusions

In this paper we have extended classical warehousing challenges to consider new classes of problems by incorporating some physical characteristics of the products into account.

Specifically, we consider that products can support at maximum weight when being transported on a lift-truck used for warehouse picking, that some products are more fragile than others, and that products belonging to different categories must be picked in a given order such as to avoid contamination.

We have adapted three classical order picking algorithms to handle the new features. Notably, we have shown how the S-shape, the largest gap, and the mid point heuristics can be used to yield very fast solutions to our problem. However, we have also shown how these solutions can be improved by proposing a very rich metaheuristic algorithm based on ALNS.

Finally, we have proved optimality for several instances and obtained better solutions and tight gaps with two mathematical models that were solved with classical and ad-hoc valid inequalities and cutting planes. We have shown that a two-index formulation outperforms a new capacity-indexed formulation for this problem. Moreover, our exact methods outperformed the solutions of the heuristic algorithms.

References

[1] K. B. Ackerman. *Practical Handbook of Warehousing.* Springer Science & Business Media, New York, 2013.

[2] M. Bortolini, M. Faccio, M. Gamberi, and R. Manzini. Diagonal cross-aisles in unit load warehouses to increase handling performance. *International Journal of Production Economics*, forthcoming, 2015.

[3] T. Chabot, L. C. Coelho, J. Renaud, and J.-F. Côté. Mathematical models, heuristic and exact method for order picking in 3d-narrow aisles. Technical Report CIRRELT-2015-18, Québec, Canada, 2015.

[4] C. Chackelson, A. Errasti, D. Ciprés, and F. Lahoz. Evaluating order picking performance trade-offs by configuring main operating strategies in a retail distributor:

a design of experiments approach. *International Journal of Production Research*, 51 (20):6097–6109, 2013.

[5] D. M.-H. Chiang, C.-P. Lin, and M.-C. Chen. The adaptive approach for storage assignment by mining data of warehouse management system for distribution centres. *Enterprise Information Systems*, 5(2):219–234, 2011.

[6] R. de Koster and E. Van der Poort. Routing orderpickers in a warehouse: a comparison between optimal and heuristic solutions. *IIE Transactions*, 30(5):469–480, 1998.

[7] R. de Koster, E. S. Van der Poort, and K. J. Roodbergen. When to apply optimal or heuristic routing of orderpickers. In B. Fleischmann, J. A. E. E. Van Nunen, M.G. Speranza, and P. Stähly, editors, *Advances in Distribution Logistics*, pages 375–401. Springer, Berlin, 1998.

[8] R. de Koster, T. Le-Duc, and K. J. Roodbergen. Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, 182 (2):481–501, 2007.

[9] J. de Vries, R. de Koster, and D. Stam. Exploring the role of picker personality in predicting picking performance with pick by voice, pick to light and RF-terminal picking. *International Journal of Production Research*, forthcoming, 2015.

[10] J. Gu, M. Goetschalckx, and L. F. McGinnis. Research on warehouse design and performance evaluation: A comprehensive review. *European Journal of Operational Research*, 203(3):539–549, 2010.

[11] R.W. Hall. Distance approximations for routing manual pickers in a warehouse. *IIE transactions*, 25(4):76–87, 1993.

[12] S. Henn, S. Koch, and G. Wäscher. Order batching in order picking warehouses: A survey of solution approaches. In R. Manzini, editor, *Warehousing in the Global Supply Chain*, pages 105–137. Springer, London, 2012.

[13] M. Hompel and T. Schmidt. *Warehouse Management: Automation and Organisation of Warehouse and Order Picking Systems.* Springer, Berlin, 2006.

[14] L. Junqueira, R. Morabito, and D. S. Yamashita. Three-dimensional container loading models with cargo stability and load bearing constraints. *Computers & Operations Research*, 39(1):74–85, 2012.

[15] R. Lahyani, L. C. Coelho, and J. Renaud. Alternative formulations and improved bounds for the multi-depot fleet size and mix vehicle routing. Technical Report CIRRELT-2015-36, Québec, Canada, 2015.

[16] W. Lu, D. McFarlane, V. Giannikas, and Q. Zhang. An algorithm for dynamic order-picking in warehouse operations. *European Journal of Operational Research*, forthcoming, 2015.

[17] J. Lysgaard. *CVRPSEP: A package of separation routines for the capacitated vehicle routing problem.* Department of Management Science and Logistics, Aarhus School of Business, Denmark, 2003.

[18] R. Manzini. *Warehousing in the Global Supply Chain.* Springer, London, 2012.

[19] M. Matusiak. *Optimizing Warehouse Order Batching when Routing Is Precedence Constrained and Pickers Have Varying Skills.* Ph.D. dissertation, Aalto University, Helsinki, 2014.

[20] M. Matusiak, R. de Koster, L. Kroon, and J. Saarinen. A fast simulated annealing method for batching precedence-constrained customer orders in a warehouse. *European Journal of Operational Research*, 236(3):968–977, 2014.

[21] M. W. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33(1):60–100, 1991.

[22] A. Pessoa, E. Uchoa, and M. V. S. Poggi de Aragão. A robust branch-cut-and-price algorithm for the heterogeneous fleet vehicle routing problem. *Networks*, 54(4): 167–177, 2009.

[23] C. G. Petersen. An evaluation of order picking routing policies. *International Journal of Operations and Production Management*, 17(11):1098–1111, 1997.

[24] C. G. Petersen and G. R. Aase. A comparison of picking, storage, and routing policies in manual order picking. *International Journal of Production Economics*, 92 (1):11–19, 2004.

[25] C. G. Petersen, G. R. Aase, and D. R. Heiser. Improving order-picking performance through the implementation of class-based storage. *International Journal of Physical Distribution & Logistics Management*, 34(7):534–544, 2004.

[26] J. C. Picard and M. Queyranne. The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. *Operations Research*, 26(1):86–110, 1978.

[27] M. V. S. Poggi de Aragão and E. Uchoa. New exact algorithms for the capacitated vehicle routing problem. In P. Toth and D. Vigo, editors, *Vehicle Routing: Problems, Methods, and Applications*, pages 59–86. MOS-SIAM Series on Optimization, Philadelphia, 2014.

[28] J.-Y. Potvin and J.-M. Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331–340, 1993.

[29] S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006.

[30] P. Shaw. A new local search algorithm providing high quality solutions to vehicle routing problems. Technical report, APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK, 1997.

[31] J. A. Tompkins, J. A. White, Y. A. Bozer, and J. M. A. Tanchoco. *Facilities Planning*. John Wiley & Sons, New York, 2010.

[32] P. Toth and D. Vigo. The family of vehicle routing problem. In P. Toth and D. Vigo, editors, *Vehicle Routing: Problems, Methods, and Applications*, pages 1–23. MOS-SIAM Series on Optimization, Philadelphia, 2014.

[33] G. Wäscher. Order picking: a survey of planning problems and methods. In H. Dyckhoff, R. Lackes, and J. Reese, editors, *Supply Chain Management and Reverse Logistics*, pages 323–347. Springer, Berlin, 2004.